

A coiled black shoestring with metal ferrules at both ends. The shoestring is made of a thick, braided black material, likely cotton or polyester, and is coiled into a loose, somewhat circular shape. The metal ferrules are silver-colored and cylindrical, one at each end of the shoestring. The background is a plain, light-colored surface.

Building Wikipedia

Scalable LAMP on a shoestring budget

Brion Vibber

GatorJUG 2007-09-12

Wikipedia is happy to
serve you...

- 7,000,000,000 page views per month
- 32,000 HTTP objects/second at peak
- Hardware budget to date: \$1 million
- Tech department staff: 4

That's not a big budget,
dude.

Web 2.0 model

1. Make a cool site
2. Slap up some ads to look like you have a business plan
3. Sell out to Google, Y!, or Microsoft
cha-ching!

Wikipedia model

1. Make a cool site
2. Incorporate as a not-for-profit, keep the site non-commercial and ad-free
3. Beg for money

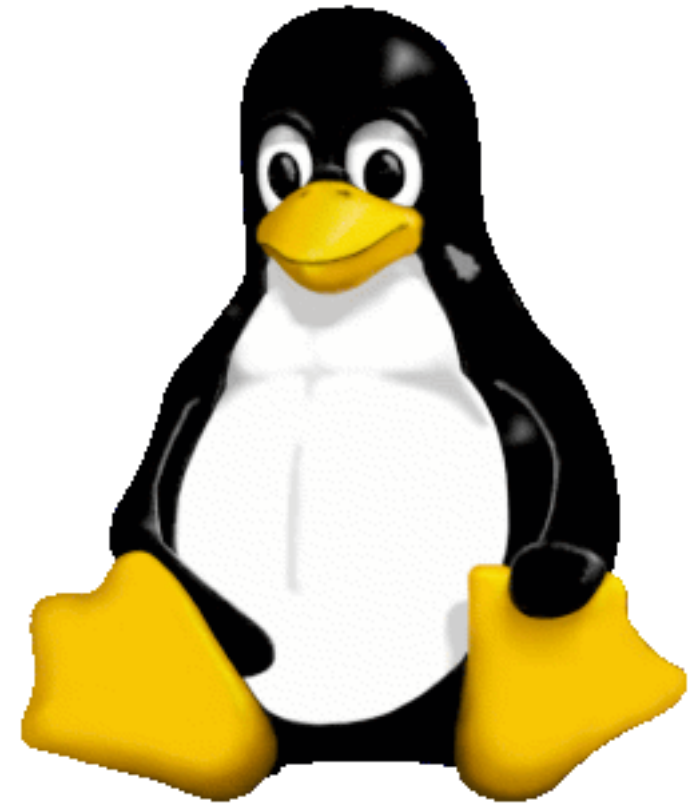
Hey, at least we're honest about it!

Wikimedia Foundation, Inc.

- 501(c)3 not-for-profit
- Funded primarily through donations, tax-deductible in the US (*hint hint*)
- Annual donation drive in Fall/Winter (*hmmm, that's coming up isn't it?*)

so when you have no money...

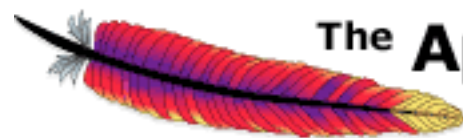
Free is good!



Free as in software!



yay :)



The **Apache Software Foundation**

<http://www.apache.org/>

Basic LAMP stack

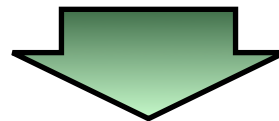
y'all know the drill...

- Linux
- Apache
- MySQL
- PHP / Perl / Python / Pwhatever

at the core...

keep it simple

Apache+PHP



MySQL

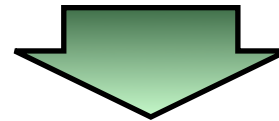
Ahhh, simple is nice.
:)

Simple is slow.

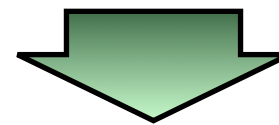
:(

First, add cache!

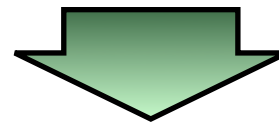
Squid



Apache+PHP+APC

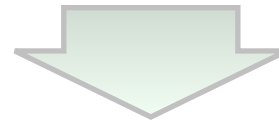


memcached



MySQL

Squid



Apache+PHP+APC

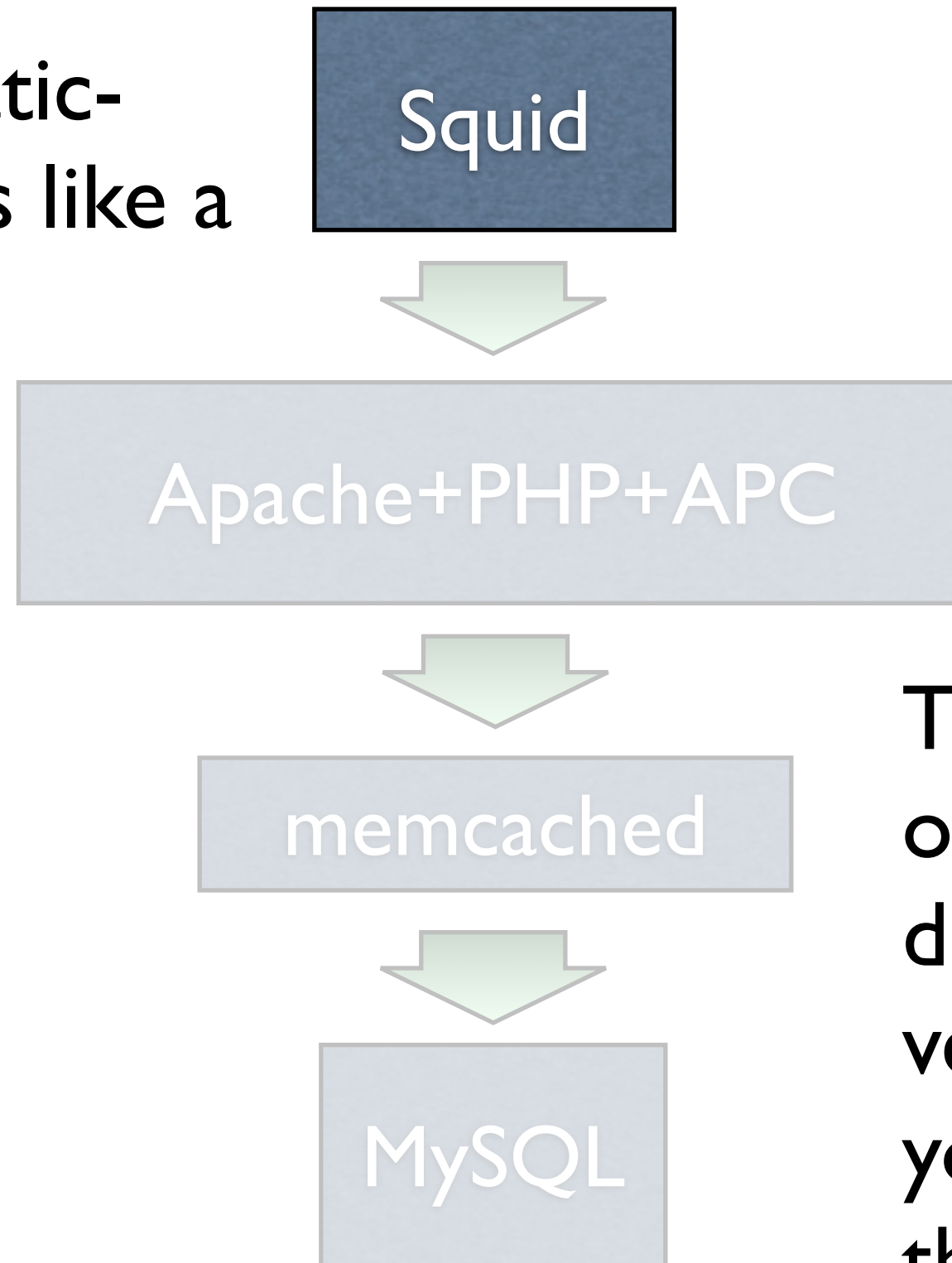


memcached



MySQL

Good for static-
dynamic sites like a
wiki...

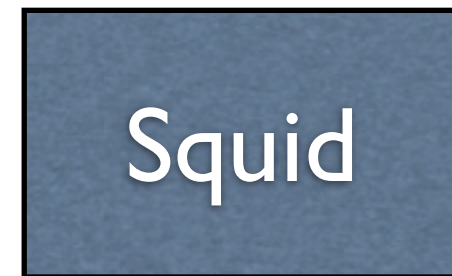


The public face
of a given page
doesn't change
very often, so
you can cache at
the HTTP level.

Seoul



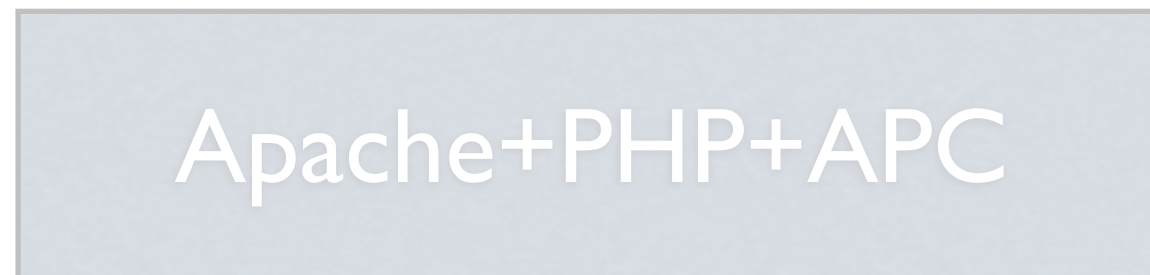
Amsterdam



Tampa

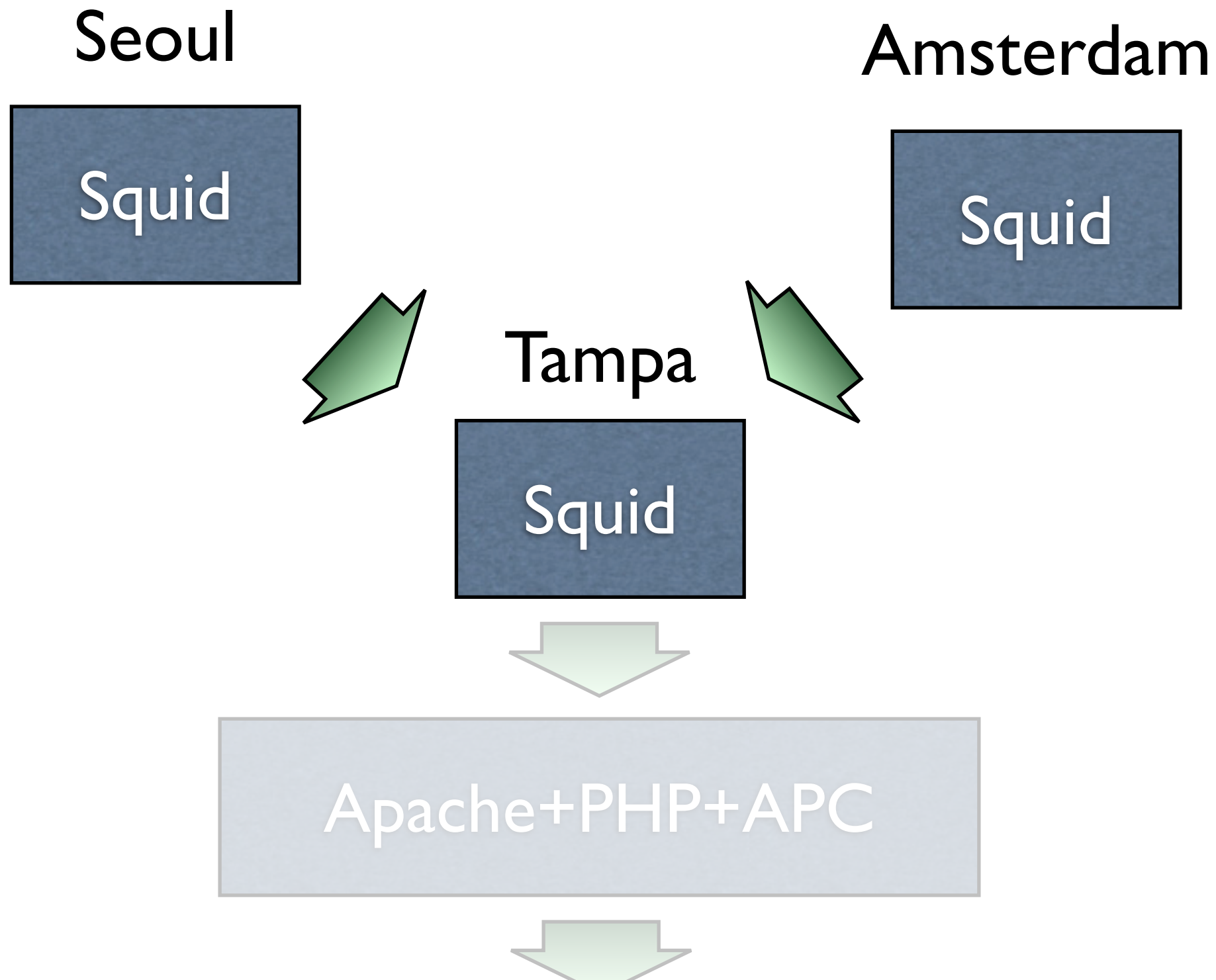


Apache+PHP+APC

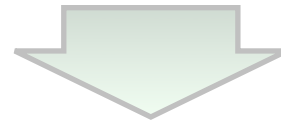


Good for geographic load balancing, too!

Use cheaper, faster local
bandwidth...



Squid



Apache+PHP+APC

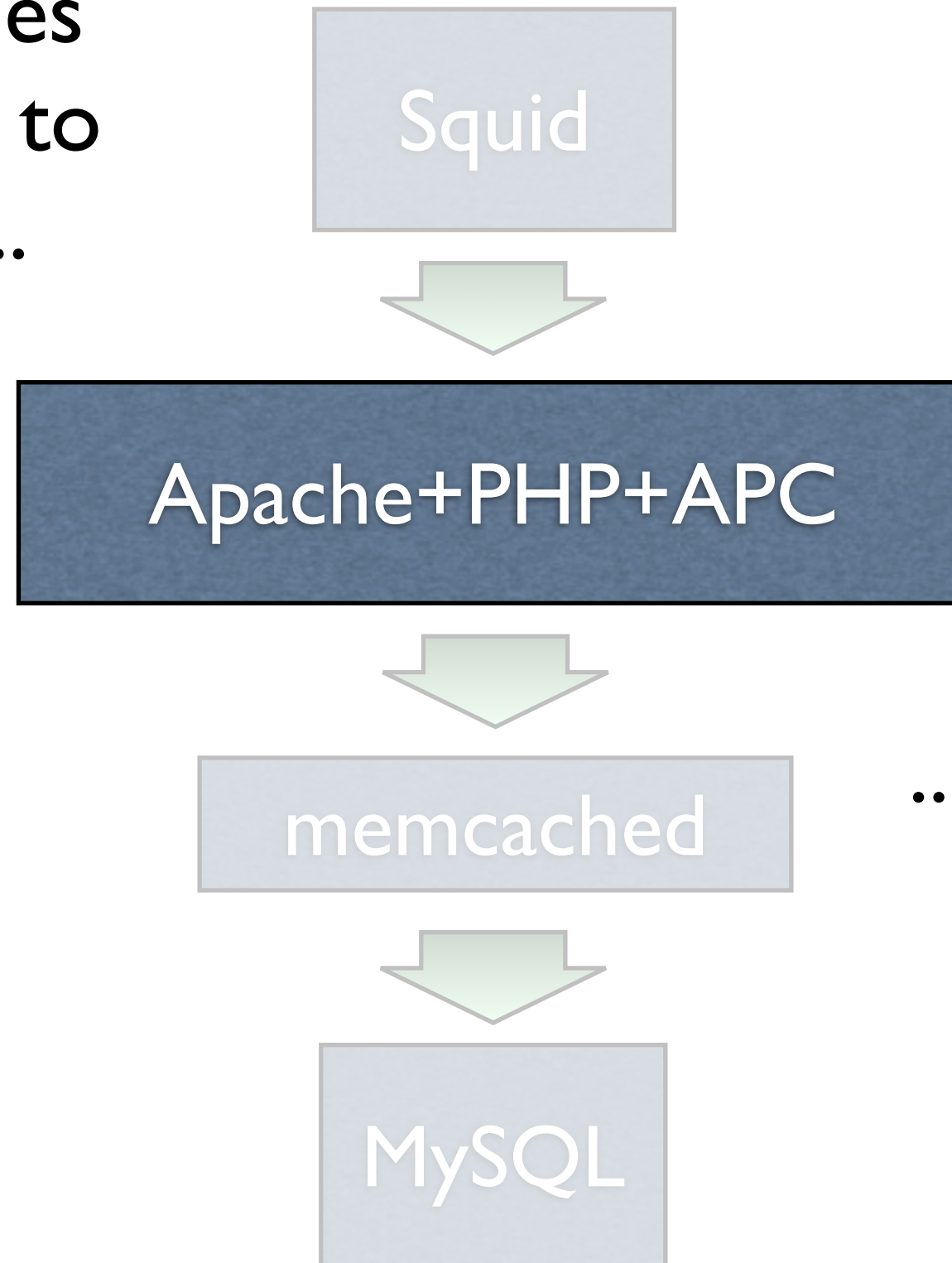


memcached



MySQL

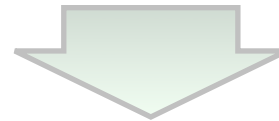
PHP compiles
your scripts to
bytecode...



...then throws it
away after
execution.

Compiling on every run adds a lot of overhead...

Squid



Apache+PHP+APC



memcached



MySQL

...especially as your application grows...

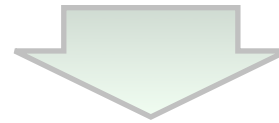


...pulling in lots of framework and library code.



Always use an opcode
cache with PHP!

Squid



Apache+PHP+APC



memcached



MySQL

Drastically reduces
startup time for
large apps, and
moderate
improvements even
for small scripts.

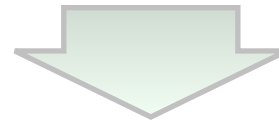
APC

eAccelerator

Zend Platform

...

Squid



Apache+PHP+APC

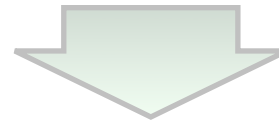


memcached



MySQL

Squid



Apache+PHP+APC



memcached

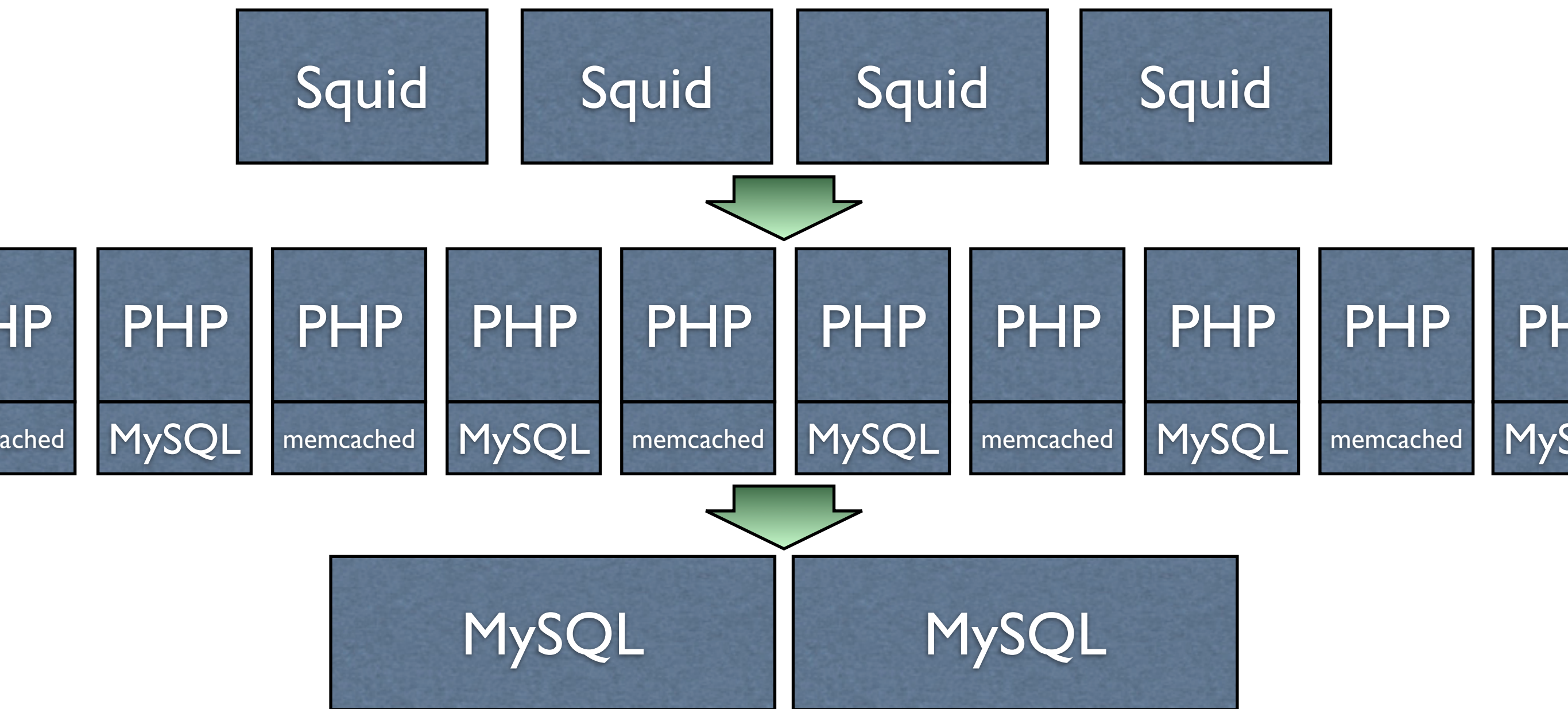


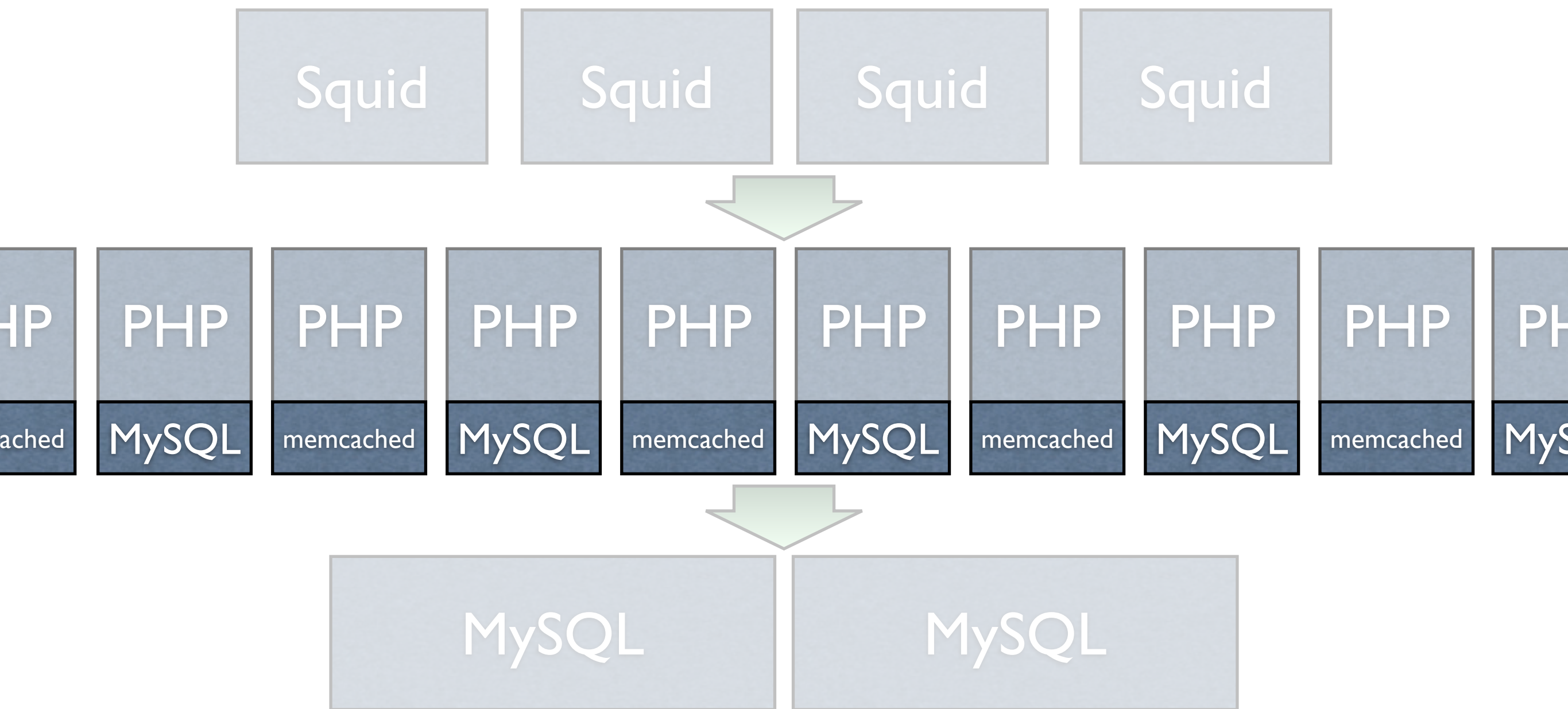
MySQL

Share temporary
data in your
network's
memory

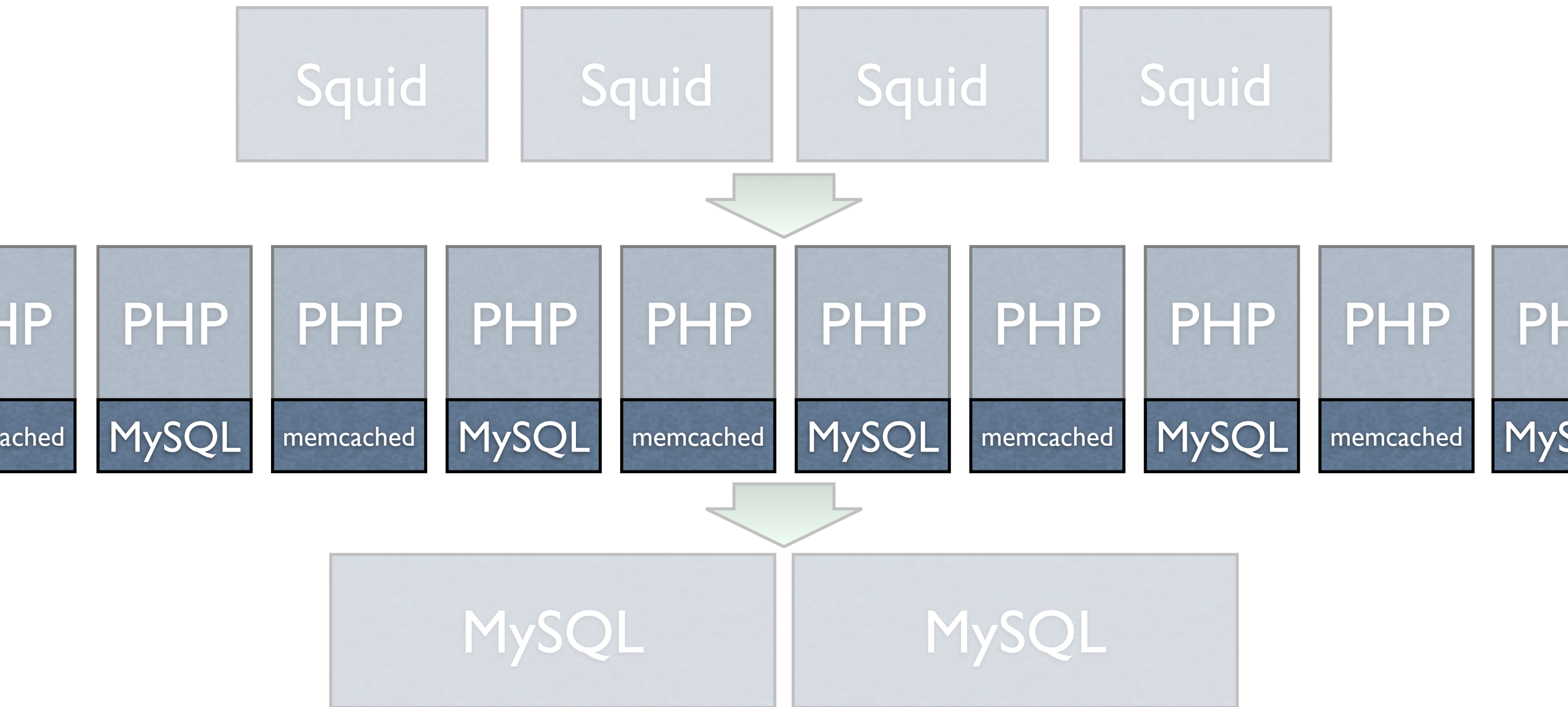
Faster than
disk-backed
database
when you
just need an
object
cache...

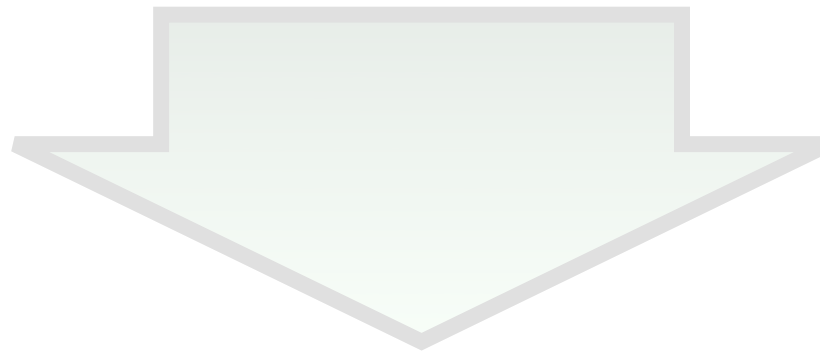
Now add cash!



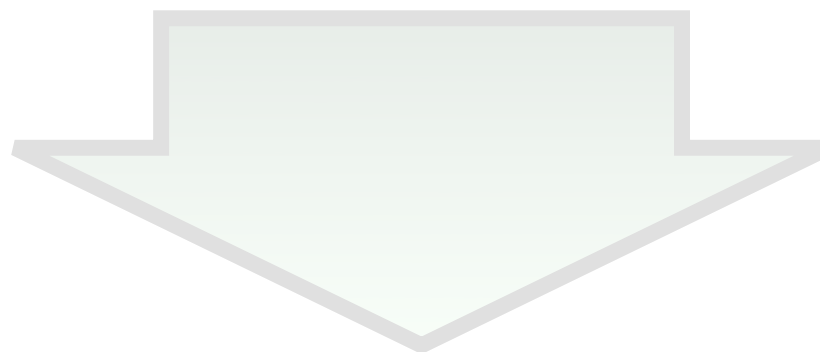
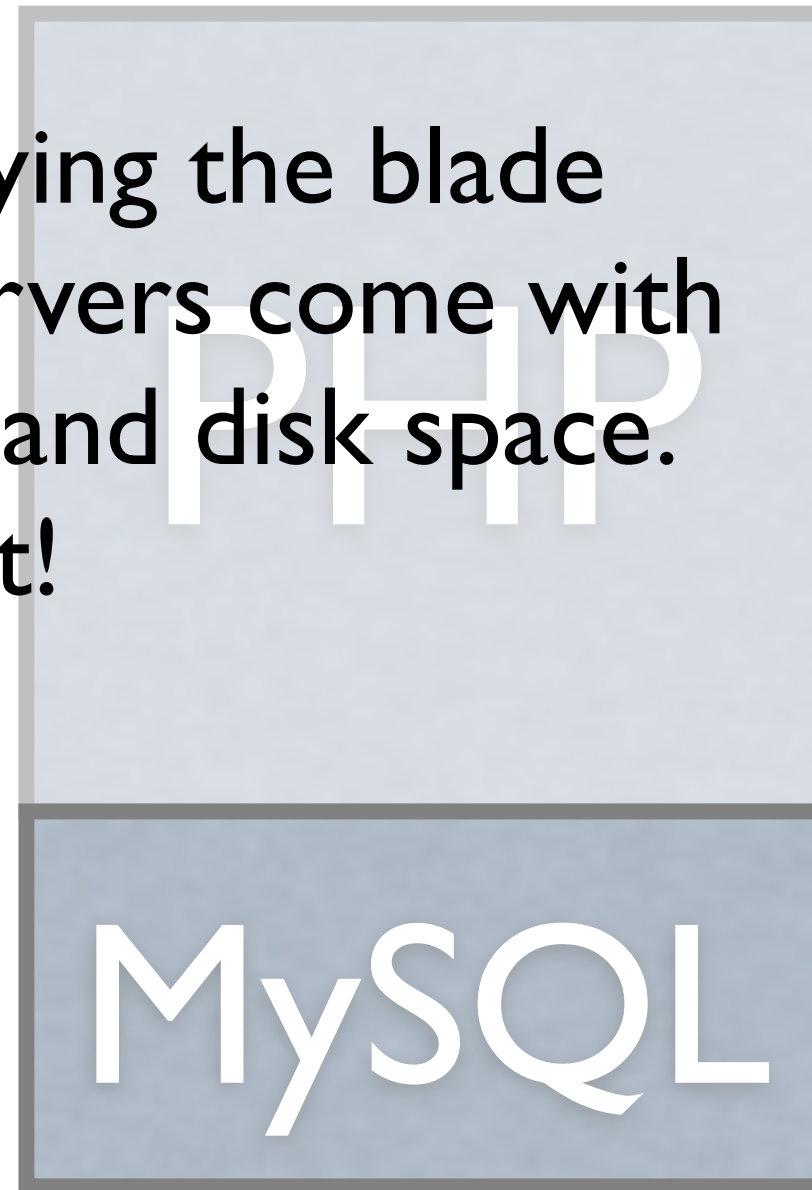
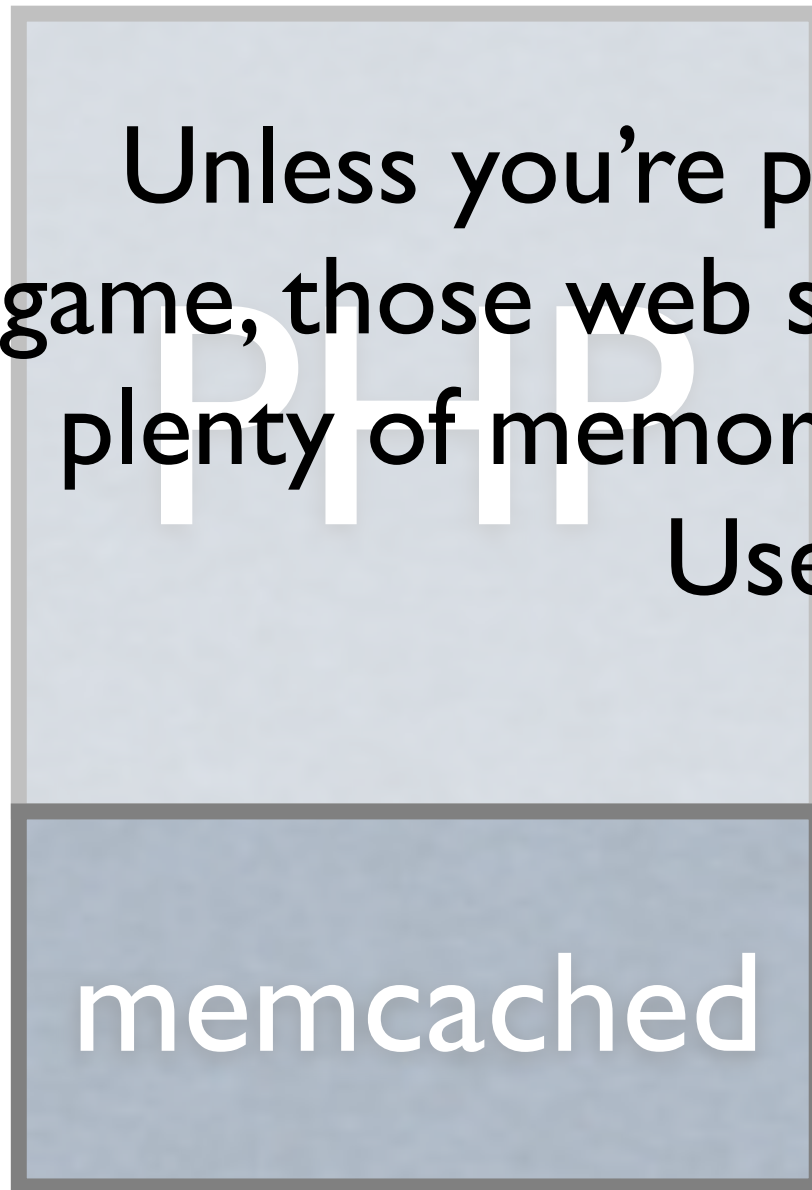


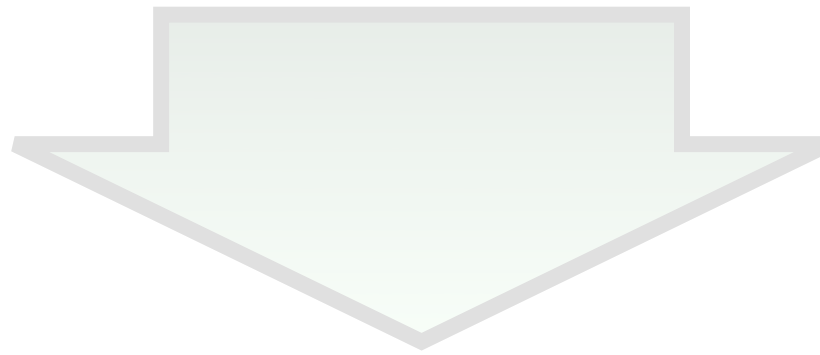
Put underutilized memory and disk to work!





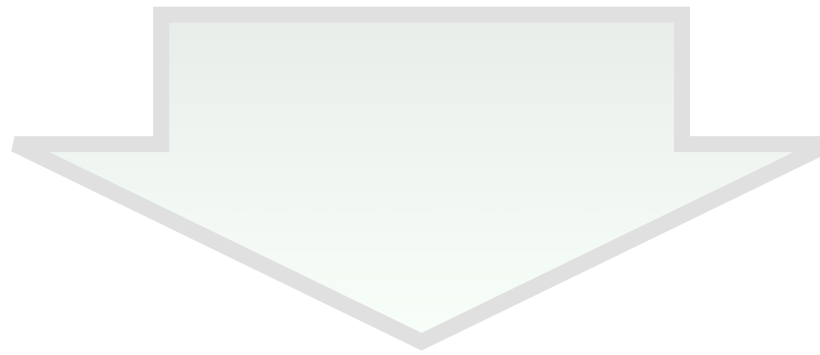
Unless you're playing the blade
game, those web servers come with
plenty of memory and disk space.
Use it!



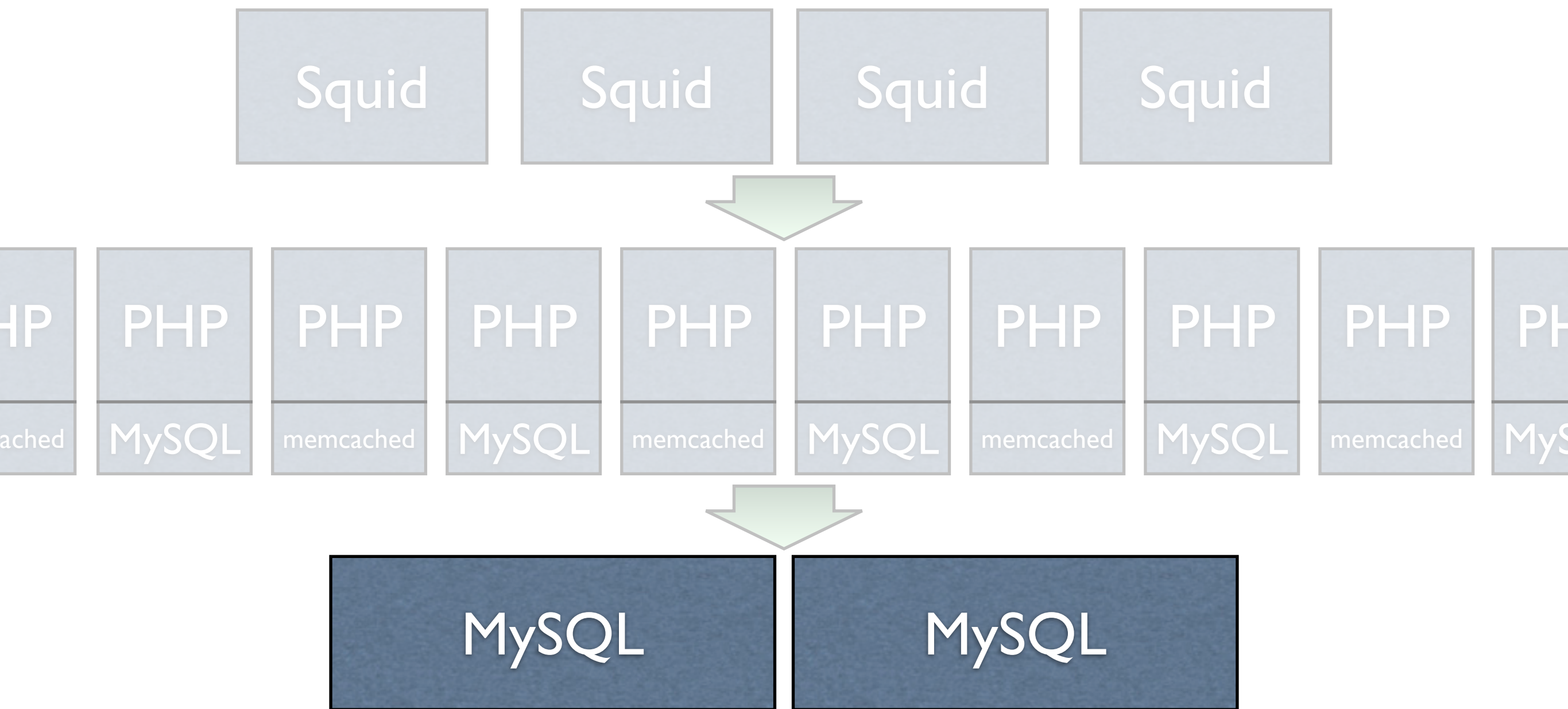


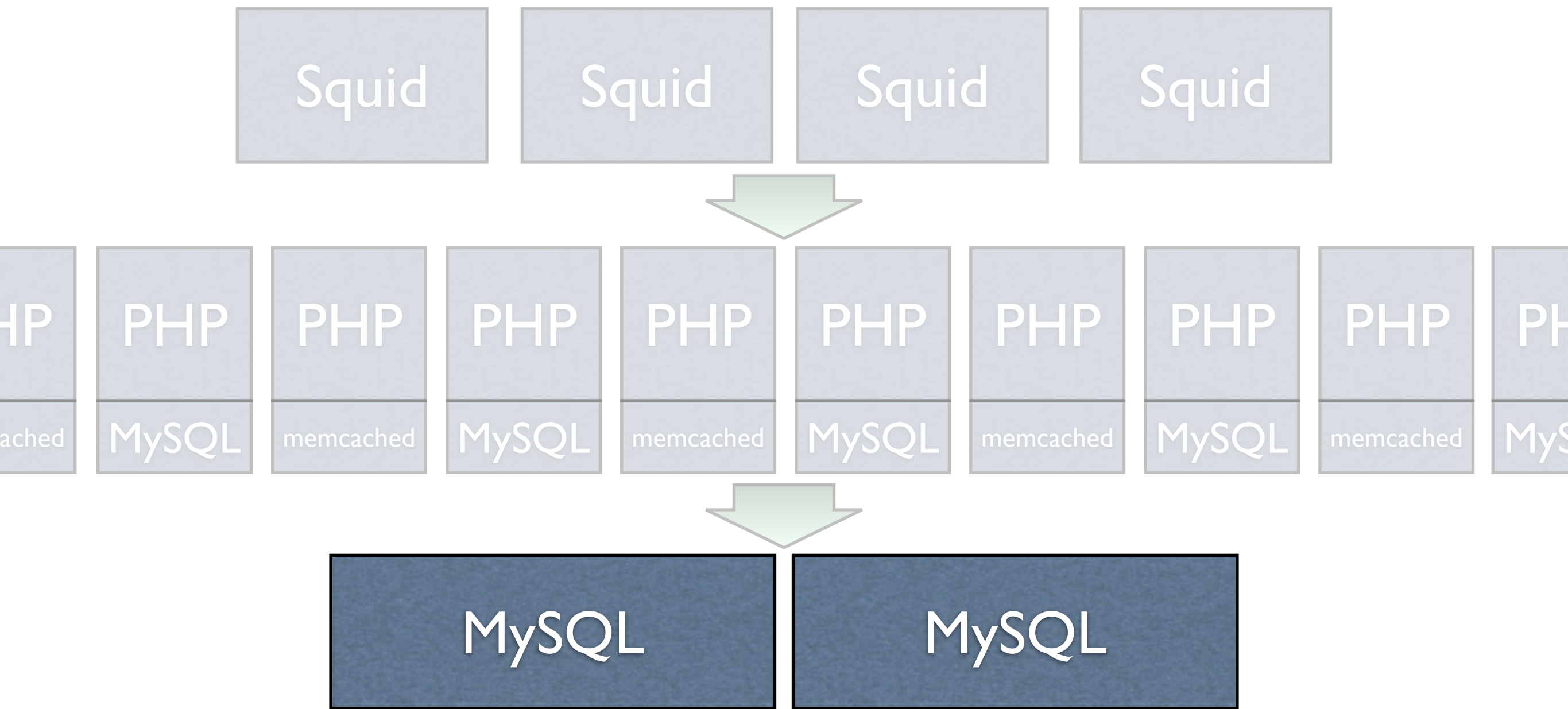
A bit of memory on
each server adds up
to a big memcached
store space...





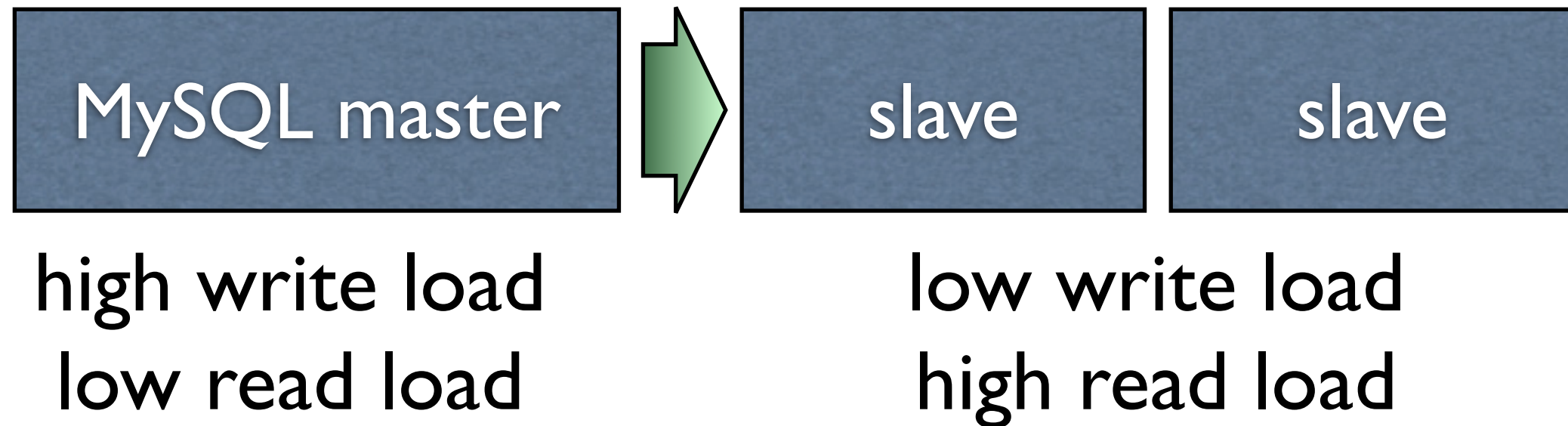
used for replicated
bulk data storage at
very little CPU cost.





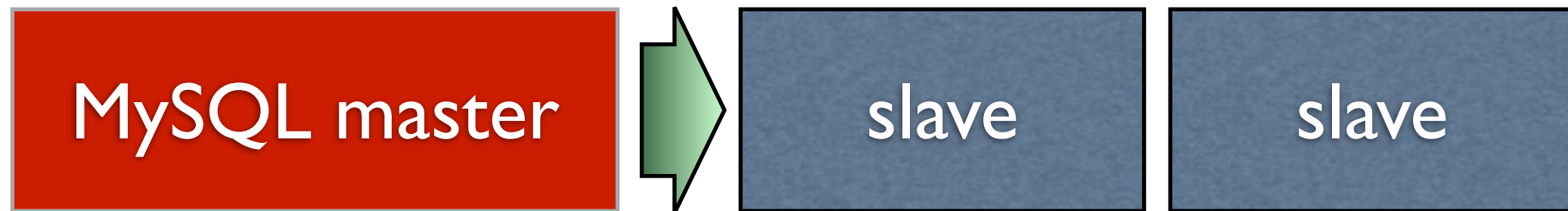
Break it up...

Replicate for speed!



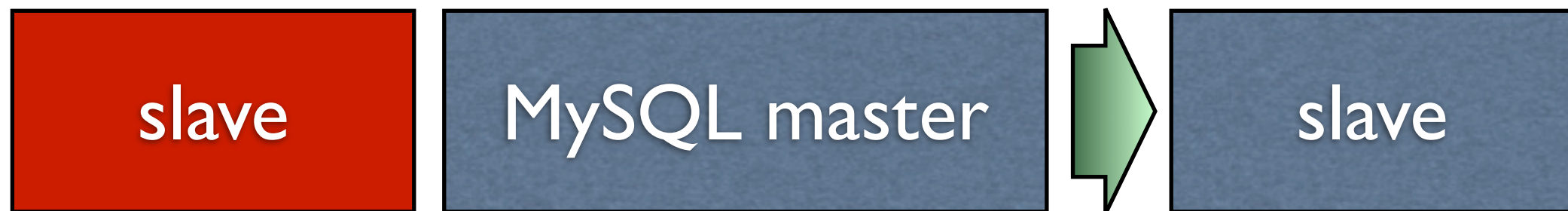
...but your application now has to think about replication lag.

& reliability...



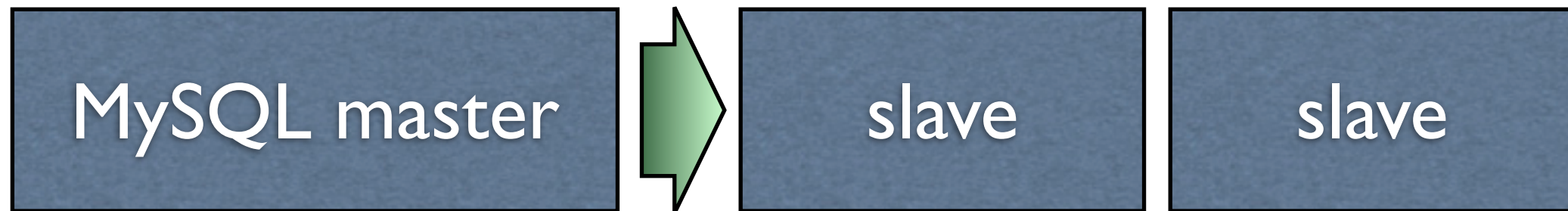
Master dead?

Promote a slave!



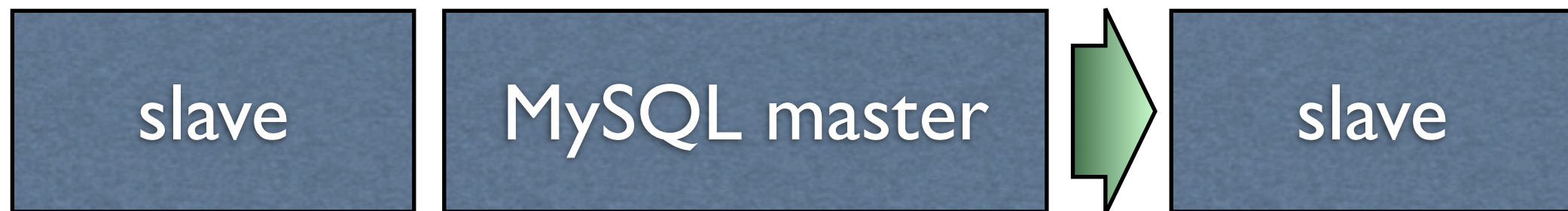
...but failover isn't automated with MySQL.

& some tricks.



apply schema changes on slaves...

...then swap masters...



...for low-downtime column and
index changes!

Data too big? Load too high?

Split along logical data partitions, such as subsites that don't interact closely.

MySQL group s1

English-language Wikipedia

MySQL group s2

Next 19 biggest wikis

MySQL group s3

Next 730 wikis

Data too big? Load too high?

Split along functional boundaries...

MySQL big iron

Page metadata, links, users...

...read/write/update

...active index scans

PHP

PHP

PHP

MySQL

MySQL

MySQL

Append-only bulk text

...nice simple blobs

General scalability tricks...

- Smart caching can keep most load away from the backend
- Keep data sets small -- look for places to spread out horizontally
- Keep worst cases fast, not just average cases

Questions?

wikimediafoundation.org